CiteSeer Find: profile recompiled cache instruction Documents Citations

Searching for PHRASE **profile recompiled cache instruction**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) CSB DBLP
No documents match Boolean query. Trying non-Boolean relevance query.
500 documents found. **Order: relevance to query.**

Instruction Cache Effects of Different Code Reordering Algorithms - Lee (1994) (Correct) (4 citations)
the software side, static code reordering based on **profile** and static code analysis can decrease the miss
**Instruction Cache** Effects of Different Code Reordering Algorithms
www.cs.washington.edu/homes/dlee/mypapers/quals.ps

Wrong-Path Instruction Prefetching - Pierce, Mudge (1994) (Correct) (25 citations)
direction of every conditional branch. The **profile** algorithm, PROF-1, will be explained later. It
a grant from the Intel Corp. Abstract **Instruction cache** misses can severely limit the performance of both
www.eecs.umich.edu/techreports/cse/1994/CSE-TR-222-94.ps.gz

Predicting Instruction Cache Behavior - Mueller, Whalley, Harmon (1993) (Correct) (16 citations)
Predicting **Instruction Cache** Behavior Frank Mueller, David B. Whalley Marion
www.cis.famu.edu/~harmon/sigplan.ps

Optimizing Instruction Cache Performance for Operating.. - Torrellas, Xia, Daigle (1995) (Correct) (36 citations)
**cache** conflicts. McFarling's technique [11] uses a **profile** of the conditional, loop, and routine structure
however, conflicts vary from operating system **recompil**ation to **recompil**ation. Therefore, ad hoc
Optimizing **Instruction Cache** Performance for Operating System Intensive
ftp.cs.uiuc.edu/pub/research-groups/csrd/iacoma/osplace.ps

Real-Time Debugging by Minimal Hardware Simulation - Mueller, Whalley, Harmon (1994) (Correct)
a prototype of the POSIX "minimal realtime system **profile**"In IEEE Workshop on Real-Time Operating
processor cycle accounting. These aspects include **cache** simulation, **instruction** frequency accounting, and
to processor cycle accounting and includes **instruction** caching, **instruction** frequency accounting, and
www.informatik.hu-berlin.de/~mueller/ftp/pub/whalley/papers/pearl94.ps.Z

Timing Predictions for Multi-Level Caches - Mueller (1997) (Correct) (7 citations)
Timing Predictions for Multi-Level **Caches** Frank Mueller Humboldt-Universitat zu Berlin,
www.informatik.hu-berlin.de/~mueller/ftp/pub/mueller/papers/lctrts97.ps.gz

Fast Prolog with an Extended General Purpose Architecture - Holmer, Sano, Carlton.. (1994) (Correct) (6 citations)
on Computer Architecture 2 32 32 shadow data **cache instruction cache** control address address
of this paper is on our architecture and **instruction** set. The costs and benefits of the special
of the special architectural features and **instructions** are analyzed. Simulated performance results
www.info.ucl.ac.be/people/PVR/bam.ps

An Accurate Instruction Cache Analysis Technique for.. - Lim, Min, Lee, Park, .. (1994) (Correct) (3 citations)
Applications, April 1994. An Accurate **Instruction Cache** Analysis Technique for Real-time Systems
archi.snu.ac.kr/PUBLICATIONS/papers/real-time/ssiim-wart-1994.ps.gz

Compiler and Hardware Support for Automatic Instruction.. - Mowry, Luk (1998) (Correct) (1 citation)
Ontario, Canada, M5S 3G4. Abstract **Instruction cache** miss latency is becoming an increasingly
reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-140.ps

Performance Enhancement of Desktop Multimedia with Multithreaded.. - Pontius (1998) (Correct)
44 5.3.2. **Profile**
.8 2.2.1. **Cache** Effects
www.eng.uci.edu/comp.arch/papers/../multithreading/mpthesis/mpthesis.ps

Design and Performance Evaluation of a Cache Assist to implement.. - John (1997) (Correct) (6 citations)
Design and Performance Evaluation of a **Cache** Assist to implement Selective Caching L. John
www.ece.utexas.edu/~ljohn/annex.ps

Procedure Mapping Using Static Call Graph Estimation - Hashemi, Kaeli, Calder (1997) (Correct) (1 citation)
of **cache** line conflicts. Most of these schemes use **profile** data in order to reposition the code in the

required to first **profile** the program, and then **recompile** using the **profile**. It can also be hard to find
grow, it becomes increasingly important to exploit **cache** memory effectively. One technique used by
www-cse.ucsd.edu/users/calder/papers/ICCA97.ps.Z

Streamlining Data Cache Access with Fast Address Calculation - Austin, Pnevmatikatos, Sohi (1995) (Correct) (21 citations)
examining the reference behavior of programs. We **profiled** the load **instructions** of several benchmarks
Streamlining Data **Cache** Access with Fast Address Calculation Todd M.
ftp.cs.wisc.edu/sohi/papers/1995/isca.fast.ps.gz

Caching in on Sisal: Cache Performance of Sisal vs. Fortran - Nico Park (Correct)
"O" level of optimization was used. The pixie **profiler** modifies the executable image so that, while
Caching in on Sisal: **Cache** Performance of Sisal vs. Fortran P. L. Nico A.
**caches** the performance is equivalent. With split **instruction** and data **caches**, performance is still
elysium.cs.ucdavis.edu/~nico/publications/sisal93.ps

Profile-Driven Instruction Level Parallel Scheduling with.. - Chekuri Dept (1996) (Correct) (10 citations)
**Profile**-Driven **Instruction** Level Parallel Scheduling
performance in the face of fixed **instruction cache** sizes. In light of this, the threshold and the
**Profile**-Driven **Instruction** Level Parallel Scheduling with Application to
theory.stanford.edu/~chekuri/postscript/micro96.ps.gz

A Quantitative Analysis of Instruction Prefetching - Kim, Min, Kim (1995) (Correct)
on misses prefetches the next block only on a **cache** miss. Tagged prefetch, an enhancement of
301-307, Sep. 1995. A Quantitative Analysis of **Instruction** Prefetching S. B. Kim S. L. Min C. S. Kim
This paper examines the interactions between **instruction** prefetching and the memory system from the
net.snu.ac.kr/archi/PUBLICATIONS/papers/cpu-cache/sbkim-euro-1995.ps.gz

Instruction Cache Fetch Policies for Speculative Execution - Lee, Baer, Calder, Grunwald (1995) (Correct) (12 citations)
miss requests, and software techniques, like **profile** driven basic-block reordering, will
**Instruction Cache** Fetch Policies for Speculative Execution Dennis
www.cse.ucsd.edu/~calder/papers/ISCA-95-Spec.ps.Z

Computer Design Strategy for MCM-D/Flip-Chip Technology - Franzon, al. (Correct)
Thus there is tremendous advantage to building the **caches** in a computer in an SRAM process and using an MCM
to build a `MegaChip' CPU consisting of an **Instruction** Fetch Unit and Execution Unit. By building
Unit and Execution Unit. By building part of the **Instruction** Fetch Unit in an optimized SRAM process,
www.ece.ncsu.edu/info.ece/vlsi_info/techreports/NCSU-ERL-96-03.PS.Z

Reducing Memory Latency via Non-blocking and Prefetching Caches - Chen, Baer (1992) (Correct) (81 citations)
Memory Latency via Non-blocking and Prefetching **Caches** Tien-Fu Chen and Jean-Loup Baer Technical Report
ftp.cs.washington.edu/tr/1992/06/UW-CSE-92-06-03.PS.Z

*First 20 documents* Next 20

Try your query at: Google (CiteSeer) Google (Web) CSB DBLP

CiteSeer.IST - Copyright NEC and IST

113 citations found. Retrieving documents...

**S. McFarling. *Program optimization for instruction caches.* In 3rd International Conference on Architectural Support for Programming Languages and Operating Systems, volume 24, pp. 183–191, Boston, MA, April 1989.**



This paper is cited in the following contexts:

*First 50 documents* Next 50

---

Code Placement using Temporal Profile Information - Gloy (1998) (2 citations) (Correct)

....with the goal of more efficiently using the instruction cache. **Several compile time code placement techniques have been proposed that use heuristics and profile information to reduce the number of conflict misses in the primary (firstlevel or L1) instruction cache by reordering the program code** [3, 18, 26, 27, 36]. Most of this work uses cache parameters such as cache size and line size as well as procedure sizes to accurately model the cache mapping of the code. **The** code placement algorithms typically use some kind of profile information to find a cache mapping that reduces cache conflict misses. **These** ....

....use a WCG, which contains only information about pairs of procedures connected by a direct call and no information about the temporal ordering of procedure calls. **As** an example, Figure 2 shows two programs that result in the same WCG but have substantially different temporal behavior. **McFarling [26] uses profile data that incorporates loop counts and probabilities for conditionals, but still retains the limitations mentioned above.** Basic block transitions, used by Torrellas et al. 36] share these limitations. **Our** technique is based on a profiling scheme that captures important information ....

[Article contains additional citation context not shown here]

S. McFarling, "*Program optimization for instruction caches.*" Proc. ASPLOS-III: 3rd Intl. Conf. on Architectural Support for Programming Languages and Operating Systems, p.183, April 1989.

---

Working Sets at Function Level - Batchu, Levy (1998) (1 citation) (Correct)

....the working set size and cache con icts. **There** has been prior work in changing code layout at the function level at compile time as well as dynamically. **There** have also been e orts to exploit program locality dynamically at other levels of granularity. **Pettis and Hanson [20] Scott McFarling [18], Hat eld and Gerald [11] Gloy and Smith [15] have presented 3 methods to rearrange the procedures, which comprise the executable, based on pro le data to improve memory locality.** Most of these use pro le data in the form of a weighted call graph (WCG) In a WCG, there is a node for each ....

Scott McFarling. *Program optimization for instruction caches.* In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems, pages 183{ 191. ACM, 1989.

---

A Study of Program Behavior to Establish Temporal Locality at .. - Levy, Murdocca (2001) (2 citations) (Correct)

....the working set size and cache con icts. **There** has been prior work in changing code layout at the function level at compile time as well as dynamically. **There** have also been e orts to exploit program locality dynamically at other levels of granularity. **Pettis and Hanson [21] Scott McFarling [19], Hat eld and Gerald [13] Gloy and Smith [15] have presented methods to rearrange the procedures, which comprise the executable, based on pro le data to improve memory locality.** Most of these use pro le data in the form of a weighted call graph (WCG) In a WCG, there is a node for each ....

Scott McFarling. *Program optimization for instruction caches.* In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems, pages 183-191. ACM, 1989.

---

Profile Guided Compiler Optimizations - Gupta, Mehofer, Zhang (2002) (1 citation) (Correct)

....in which blocks b and d are packed into the same cache line. **The** number of cache misses is greatly reduced in this case. **Here** we have illustrated the application of code layout optimization at the basic block level. **Techniques for layout optimization at procedural level have also been developed** [29]. 5. **CONCLUDING** REMARKS In this chapter we have identi ed optimization opportunities that may exist during program execution but cannot be exploited without the availability of pro le data. **Di** erent types of pro le data that are useful for code optimization were identi ed. **The** use of this pro le ....

S. McFarling, *Program Optimization for Instruction Caches,* The 3th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 183-191, April 1989.

Software-assisted Cache Replacement Mechanisms for.. - Jain, Devadas, Engels, .. (2001) (2 citations) (Correct)

....a varied set of methods for automatic cache control instruction insertion. 2.2 Memory Exploration in Embedded Systems Cache memory issues have been studied in the context of embedded systems. **McFarling presents techniques of code placement in main memory to maximize instruction cache hit ratio [10, 14].** A model for partitioning an instruction cache among multiple processes has been presented [7] Panda, Dutt and Nicolau present techniques for partitioning on chip memory into scratchpad memory and cache [12] The presented algorithm assumes a fixed amount of scratchpad memory and a fixed size ....

S. McFarling. *Program Optimization for Instruction Caches*. In Proceedings of the 3 rd Int'l Conference on Architectural Support for Programming Languages and Operating Systems, pages 183--191, April 1989.

Application-Driven Synthesis of Memory-Intensive.. - Kirovski, Lee.. (1999) (Correct)

.... **OF THE CACHE AREA AND LATENCY MODEL: A AREA, L LATENCY [13] conflict misses in large direct mapped instruction caches has been proposed [3] Static code repositioning by using cache line coloring at the procedure or basic block level has been an alternative approach proposed and evaluated in [12], 13] and [27] Similar technique for profile driven data repositioning has been proposed in [26] III.** PERFORMANCE MODELING In this section, we describe the hardware performance models for caches and processor cores. **Three** factors combine to influence system performance: cache miss rates, ....

S. McFarling, *"Program optimization for instruction caches,"* In Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems, 1989, pp. 183--191.

Static Profiling - Ihle (1994) (Correct)

....this work is to extend their work to static profiling. **The** advantages of the availability of static profiling are quite obvious. **Many parts of optimizing compilers rely on profile data to perform good optimizations, for example trace scheduling [Fis81] register allocation [Wal86] and code motion [McF89].** In general these optimizations take advantage of locating those 10 of the program code, in which 90 of the run time is spent. **Up** to now it is common practice to get profile information by running a program and measuring the interesting data (e.g. block counts) by an appropriate tool like prof, ....

Scott McFarling. *Program optimization for instruction caches*. In Third International Symposium on Architectural Support for Programming Languages and Operating Systems, April 1989. Published as Computer Architecture News 17(2).

Code Layout Optimizations for Transaction Processing.. - Ramírez.. (2001) (2 citations) (Correct)

....TPC C benchmarks on AlphaServers. 6 Discussion and Related Work Code layout optimizations were originally proposed to reduce the working set size of applications for virtual memory [8, 11, 10] More recent work has focused on the reduction of branch mispredicts and cache misses. **McFarling [18] describes an algorithm that uses the loop and call structure of a program to determine which parts of the program should overlay each other in the cache and which parts should be assigned to non conflicting addresses.** Hwu and Chang [13] describe a profile based algorithm which uses function ....

S. McFarling. *Program optimization for instruction caches*. Proceedings of the 3rd Intl. Conferenceon Architectural Support for Programming Languages and Operating Systems, pages 183--191, Apr. 1989.

Automated Design of Finite State Machine Predictors for.. - Sherwood, Calder (2001) (Correct)

....FSM predictors are used in a few areas of computer architecture, and summarize initial results for using automated FSM predictors to guide confidence estimation used to guide value prediction. 6. **1 Cache Management Cache management schemes have been proposed that perform intelligent replacement [16], cache exclusion [29] and they use a small FSM counter to determine when the optimization should be applied.** In addition, prefetching architectures have used FSM predictors to determine when to initiate prefetching for a load and to guide stream buffer allocation [25] 6.2 Power Control Manne ....

S. McFarling. *Program optimization for instruction caches*. In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III), pages 183--191, April 1989.

Design of Trace Caches for High Bandwidth Instruction Fetching - Sung (Correct)

....of the mispredict recovery time. **These** two factors have long been identified as fetch performance bottlenecks and are relatively well researched topics. **The design of instruction caches has been studied in great detail in order to lessen the impact of instruction cache misses on fetch bandwidth [31] [36] Likewise, there have been many studies done to improve branch prediction accuracy [16] 25] To date, the techniques developed to reduce instruction cache misses and increase branch prediction accuracy have been very successful in improving fetch bandwidth.** However, as the issue rates for ....

S. McFarling, *"Program Optimization for Instruction Caches,"* Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems, April 1989

---

Application-Specific Memory Management for Embedded.. - Chiou, Jain, Rudolph, .. (1999) (10 citations) (Correct)

....space to a small region in the cache (useful for memory mapped devices) 4.2 Memory Exploration in Embedded Systems Cache memory issues have been studied in the context of embedded systems. **McFarling presents techniques of code placement in main memory to maximize instruction cache hit ratio [8, 16].** A model for partitioning an instruction cache among multiple processes has been presented [6] Panda, Dutt and Nicolau present techniques for partitioning on chip memory into scratchpad memory and cache [11] The presented algorithm assumes a fixed amount of scratchpad memory and a fixed size ....

S. McFarling. *Program Optimization for Instruction Caches*. In Proceedings of the 3 rd Int'l Conference on Architectural Support for Programming Languages and Operating Systems, pages 183--191, April 1989.

---

Global Trade-off between Code Size and Performance for .. - Heydemann, Bodin.. (Correct)

....cache is explored. 2. **If** an outer loop does not fit in the cache there is no reuse from one iteration of this loop to the 2 next. **In** this case, innermost loops still may exhibit reuses. 3. **There** is no instruction cache interference. **This problem can be addressed independently with methods [13, 12, 25] to lay out the code such as no or few interferences occur.** We only consider the first level instruction cache. 4. **Data** cache misses are assumed invariant by loop unrolling. **We** could verify this assumption in our experiments 1 . 5. **It** is assumed that the compiler generates the code for ....

S.McFarling. *Program optimization for instruction caches*. In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III), pages 183--191, April 1989.

---

Feedback directed optimization in Compaq's compilation.. - Robert Cohn Robert (1999) (11 citations) (Correct)

....counts, is the most important, especially for integer programs. **The** second use, delaying register reuse, is more important for floating point programs where scheduling for long operation latency is important. 3. **7 Code layout Our code layout algorithm is essentially the same as Pettis and Hansen [4,5,6,9].** Its goal is to reduce instruction cache misses and improve instruction fetch by using profile information to guide the layout of code in memory. **We** found that the algo **# #** Figure 4: Function from xlisp where rarely called is useful 6 rithm worked well, except in its handling of branches for ....

S. McFarling, *"Program Optimization for Instruction Caches,"* ASPLOS III Proceedings, Boston, Mass. (April 1989): 183-193.

---

The Costs and Benefits of Cloning in a Lazy Functional Language - Faxén (2001) (Correct)

....cloned program, something that currently consumes an inordinate amount of compile time. **When we apply the profiling based code placement to larger programs, we will no doubt see its effectiveness diminished, but fortunately, several more sophisticated algorithms can be found in the literature [8, 7, 6].** We would also like to find a better heuristic method, since profile guided methods are less convenient to use. **Also**, we have not yet looked at aligning procedures at cache line boundaries which might further decrease the miss rate. ....

Scott McFarling. *Program optimization for instruction caches*. In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems, pages 183--191, 1989.

---

Non-Sequential Instruction Cache Prefetching for.. - Veidenbaum, Zhao.. (1999) (2 citations) (Correct)

....keep track of dynamic branch information. **A** two level branch predictor proposed in [YePa91] uses two levels of branch history to predict branch direction. **Hybrid branch predictors composed of several single scheme predictors and a way to select one of them at a particular time have been proposed [McFa89, ChHP95].** Instruction prefetching has been addressed in the past primarily through sequential prefetch or code layout techniques [Smit82, DEC82, SmHs92, HwCh89, McFa89, Joup90, ERPR95, UNMS95, XiTo96, LBCG95 Intel93] Sometimes instruction prefetch was initiated along both possible branch paths ....

....branch direction. **Hybrid branch predictors composed of several single scheme predictors and a way to select one of them at a particular time have been proposed [McFa89, ChHP95] Instruction prefetching has been addressed in the past primarily through sequential prefetch or code layout techniques** [Smit82, DEC82, SmHs92, HwCh89, McFa89, Joup90, ERPR95, UNMS95, XiTo96, LBCG95 Intel93]. Sometimes instruction prefetch was initiated along both possible branch paths [Intel93] Compiler assistance can help by code layout or by identifying the end of a basic block to stop prefetching [HwCh89, McFa89, XiTo96] The main improvement comes from adding a sequential prefetcher as has been ....

[Article contains additional citation context not shown here]

S. McFarling, *"Program Optimization for Instruction Caches"*, International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 183-191, 1992.

---

Microarchitectural and Compile-Time Optimizations for.. - Kalamatianos (2000) (1 citation) (Correct)

.... **to store the conflicting code module in the cache [49] iii) implement a mapping function in hardware so that fewer code modules map to the same cache location [50, 51] and (iv) reorder the code modules in the main memory address space at compile time so that fewer conflicts may occur at run time** [52, 53, 54]. We pursue the last method. **We** first study the temporal interaction among procedures since accurate temporal information has not been used in the context of code reordering until recently [55] We then attempt to improve code spatial locality and instruction fetch efficiency with intraprocedural ....

....block globally. **Branch alignment is a form of basic block positioning technique that attempts to minimize the effects of 11 branch mispredictions and misfetches [69, 70, 71] Most other related work on basic block reordering has targeted improving fetch unit effectiveness and memory access time** [53, 54, 66, 72, 67]. The main idea behind all these strategies is to rearrange code units so that conflicts between them at different levels of the memory hierarchy (1st and 2nd level caches, main memory) are reduced. **In** addition, the new ordering of code should improve spatial locality and cache utilization. **We** ....

[Article contains additional citation context not shown here]

S. McFarling. *Program Optimization for Instruction Caches*. In Proceedings of the International Symposium on Architectural Support for Programming Languages and Operating Systems, pages 183--191, April 1989.

---

Unknown - The Costs And (Correct)

No context found.

S. McFarling. *Program optimization for instruction caches*. In 3rd International Conference on Architectural Support for Programming Languages and Operating Systems, volume 24, pp. 183--191, Boston, MA, April 1989.

---

Automated Design of Finite State Machine Predictors - Timothy Sherwood Brad (2001) (Correct)

No context found.

S. McFarling. *Program optimization for instruction caches*. In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS III), pages 183--191, April 1989. 29

---

Ecient and Accurate Analytical Modeling of - Whole-Program Data Cache (Correct)

No context found.

S. McFarling. *Program optimization for instruction caches*. In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'89), pages 183--191, Apr. 1989.

---

Universit□at Karlsruhe (TH) - Institut Ur Programmstrukturen (Correct)

No context found.

pp. McFarling, *\Program Optimization for instruction caches"*, Proceedings of ASPLOS III, 1989

---

Using Cache Line Coloring to Perform Aggressive Procedure - Inlining Hakan Aydn (Correct)

No context found.

S. McFarling, *"Program optimization for instruction caches,"* Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems, April 1989.

---

In Proceedings of the 35th International Symposium on.. - Compiling For.. (Correct)

No context found.

S. McFarling. *Program optimization for instruction caches.* In Third International Conference on Architectural Support for Programming Languages and Operating Systems, pages 183--191, Apr. 1989.

---

Static Branch Frequency and Program Profile Analysis - Wu, Larus (1994) (26 citations) (Correct)

No context found.

McFarling, S , *"Program Optimization for Instruction Caches,"* Third International Conference on Architectural Support for Programming Language and Operating Systems (April 1989) pp. 183- 191.

---

Trap-driven Memory Simulation - Uhlig (1995) (2 citations) (Correct)

No context found.

McFarling, S. *Program optimization for instruction caches.* In Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems, Boston, MA, ACM, 183-191, 1989.

---

Spike is a Performance Tool Developed By - Digital To Optimize (Correct)

No context found.

S. McFarling, *"Program Optimization for Instruction Caches,"* ASPLOS III Proceedings, Boston, Mass. (April 1989): 183--193.

*First 50 documents* Next 50

Online articles have much greater impact More about CiteSeer.IST Add search form to your site Submit documents Feedback

CiteSeer.IST - Copyright NEC and IST